UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/681,064	12/15/2000	Andrew L. Bliss	MSFT-0218	9482

41505 7590 03/13/2007

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2192	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	03/13/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	09/681,064	BLISS ET AL.	
	Examiner	Art Unit	
	Michael J. Yigdall	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 December 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3,7,10,11,14-20 and 27-31* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3,7,10,11,14-20 and 27-31* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is responsive to Applicant's submission filed on December 26, 2006. Claims 1, 3, 7, 10, 11, 14-20 and 27-31 are pending.

Response to Arguments

2. Applicant's arguments have been fully considered but they are not persuasive.

At the outset, Applicant is respectfully reminded that the test for obviousness is not that the claimed invention must be expressly suggested in any one or all of the references, but rather, the test is what the combined teachings of the references would have suggested to those of ordinary skill in the art. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981). In this case, the combined teachings of the You, Niemi and Hawley references would have suggested the claimed subject matter to those of ordinary skill in the art.

Applicant's claims are directed to a debugger that comprises a "debugging type abstraction" and a "processor abstraction." The abstractions describe the arrangement or organization of programming code in the debugger, as Applicant indicates (remarks, page 12). As set forth in the Office action, You teaches or suggests the processor abstraction as recited in the claims. In view of Niemi, the references further teach or suggest the debugging type abstraction as recited in the claims.

Applicant acknowledges the "debug" objects illustrated in FIG. 4 of Niemi (remarks, page 13), but contends that Niemi does not teach or suggest that the debug objects share programming code organized into a tree with leaf nodes, such as in the claimed debugging type abstraction (remarks, page 14).

However, as cited in the Office action, FIG. 4 of Niemi illustrates a class hierarchy, which is to say an abstraction. The debug objects comprise programming code and are clearly organized into a tree with leaf nodes. The "base Debug class 400 ... defines the generic behaviors of that class" and "there are a plurality of preconfigured debug objects defined from the base Debug class 400" (column 8, lines 13-21). Furthermore, as Applicant notes (remarks, page 13), the "debug sub-class 416 is used to define one or more Debug debug objects 418a-418n for use in debugging an application or process," and each "provides a different granularity of debugging information or control" (column 8, lines 37-42).

The debug objects share programming code in the ordinary sense of object-oriented inheritance. For example, the "generic behaviors" defined in base Debug class 400 are shared among all debug objects. Debug debug object 418b, for example, includes the generic behaviors of the base class and any behaviors defined in debug subclass 416, in addition to the behaviors that Debug debug object 418b itself defines corresponding to "debug level 2." Niemi thus teaches or suggests a debugging type abstraction.

Moreover, as set forth in the Office action, You teaches or suggests the claimed tree form with respect to the processor abstraction. The teachings of Niemi suggest implementing a debugging type abstraction in the same form.

Nonetheless, Applicant contends that the teachings of You are deficient with respect to the claimed tree form (remarks, page 15).

Specifically, Applicant characterizes the addressing abstraction illustrated in FIG. 9 of You such that "only a particular one of the nodes within such addressing abstraction is selected to locate a particular address, depending on operating system and/or platform," and contends, "In

contrast, the present invention as recited in claims 1 and 20 requires that the programming code for both the debugging type abstraction and the processor abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom such that each respective block is defined to include a plurality of nodes extending from the base node to a particular leaf node" (remarks, page 15).

However, as set forth in the Office action, the addressing abstraction of You is a processor abstraction. As illustrated in FIG. 9, the abstraction is organized into a tree form with generic code at a base node (e.g., TUniversalAddress) and more specific levels of code branching out at nodes therefrom (e.g., T32BitAddress). The abstraction further includes leaf nodes from which no other nodes branch out (e.g., TM68000Address). Each "processor block" is defined to include a plurality of nodes extending from the base node to a particular leaf node (e.g., the "block" defined to include TUniversalAddress, T32BitAddress and TM68000Address). You discloses that the TM68000Address node implements functions for a Motorola M68000 microprocessor and is a subclass of T32BitAddress (see, for example, column 28, lines 21-55), which is in turn derived from TUniversalAddress. Thus, the "processor block" for a Motorola M68000 microprocessor, for example, is defined to include a plurality of nodes extending from the TUniversalAddress base node to the TM68000Address leaf node.

Similarly, Applicant characterizes the tree illustrated in FIG. 9 of You such that it "has a plurality of nodes, each for a particular type of address, where selecting a particular node is necessary for selecting the corresponding type of address," and contends, "In contrast, the tree recited in claims 1 and 20 has a plurality of nodes where a particular debugging type block or

processor block is defined by selecting a plurality of such nodes extending from a base node to a particular leaf node" (remarks, page 15).

Again, however, a "processor block" in You includes a plurality of nodes extending from the TUniversalAddress base node to a particular leaf node, such as the TM68000Address leaf node in the example above. The addressing abstraction of You represents a class hierarchy (see, for example, column 27, lines 13-24). Selecting a particular leaf node implies selecting every node along that branch in the hierarchy.

Moreover, applying the same reasoning to the abstraction illustrated in FIG. 4 of Niemi, a "debugging type block" would include a plurality of nodes extending from a base node (e.g., base Debug class 400) to a particular leaf node (e.g., Debug debug object 418b). The language of the claims does not distinguish over the references.

In response to Applicant's statement that "the Examiner has provided insufficient teachings, suggestions, or motivations for combining the teachings of the cited references as proposed to arrive at the claimed invention" (remarks, page 16), Applicant's statement is merely a general allegation, and is not a persuasive argument. The examiner notes that Applicant does not present any evidence of nonobviousness.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 3, 7, 10, 11, 14, 15, 18-20 and 27-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,158,045 to You (art of record, "You") in view of U.S. Patent No. 6,470,388 to Niemi et al. (art of record, "Niemi").

With respect to claim 1 (previously presented), You discloses a debugger for debugging any of a plurality of debuggees (see, for example, the abstract), each debuggee having a processor attribute selected from a plurality of processor attributes and representative of a type of processor associated with the debuggee (see, for example, column 72, line 55 to column 73, line 5, which shows that the debuggee has processor attributes from which the architecture or the type of the processor may be determined).

You does not expressly disclose each debuggee having a debugging type attribute selected from a plurality of debugging type attributes and representative of a type of debugging to be performed with respect to the debuggee.

However, You discloses that the engine is intended for a plurality of debugging types (see, for example, column 5, lines 43-55), and discloses a plurality of debugging types (see, for example, column 6, lines 31-55). It would have been obvious to one of ordinary skill in the art at the time the invention was made for each debuggee to include a debugging type attribute, in addition to the processor attributes (see, for example, column 9, lines 17-25), so as to designate a particular debugging type. Such an addition would, for example, enable the engine to perform the designated type of debugging without the need for user input.

You further discloses that the debugger is instantiated on a computer (see, for example, FIG. 1) and comprises:

(a) an engine for performing debugging functions with respect to any of the plurality of debuggees (see, for example, column 9, lines 17-25, which shows a portable debugging engine for debugging any of a plurality of debuggees), the engine including:

(i) a plurality of debugging type blocks (see, for example, column 66, lines 62-67, which shows that the engine includes a plurality of classes or blocks to provide types of debugging services), each debugging type block for supporting at least one of the plurality of debugging type attributes (see, for example, column 6, lines 31-55, which shows a plurality of debugging types supported by the engine); and

(ii) a plurality of processor blocks (see, for example, column 66, lines 62-67, which shows that the engine includes a plurality of classes or blocks to provide debugging services for a plurality of processors), each processor block for supporting at least one of the plurality of processor attributes (see, for example, column 9, lines 17-25, which shows a plurality of processor attributes supported by the engine),

(b) wherein a particular debugging type block and a particular processor block are selected for debugging a particular debuggee based on the debugging type attribute and processor attribute of the particular debuggee (see, for example, column 4, lines 41-50 and column 5, lines 47-59, which shows selecting the classes or blocks for debugging a particular debuggee based on attributes of the debuggee),

(c) wherein the plurality of debugging type blocks are organized into a debugging type abstraction available to provide debugging type services that vary in implementation for each debugging type (see, for example, see, for example, column 5, lines 43-59, which shows that the engine is organized into abstractions to provide debugging services that vary in implementation),

(d) wherein the debugging type abstraction comprises programming code, and wherein at least a portion of the programming code for the debugging type abstraction is common as between at least some debugging type blocks and is shared by such debugging type blocks (see, for example, column 10, lines 29-40, which shows that the abstractions include common programming code that is shared and reused),

(e) wherein the programming code for the debugging type abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom, the debugging type abstraction nodes including leaf nodes from which no other nodes branch out, each debugging type block being defined to include a plurality of nodes extending from the base node to a particular leaf node (see, for example, FIG. 9 and column 27, lines 13-24, which shows that the abstractions are organized into trees with common code at the base node and derived code at the leaf nodes, and see, for example, column 28, lines 21-55, which shows one such block defined to include a plurality of nodes extending from the base node to a leaf node).

(f) wherein the plurality of processor blocks are organized into a processor abstraction available to provide processor services that vary in implementation for each processor (see, for example, see, for example, column 5, lines 43-59, which shows that the engine is organized into abstractions to provide debugging services that vary in implementation, and see, for example, FIG. 9 and column 27, lines 1-7, which shows a processor abstraction to provide memory addressing for each processor),

(g) wherein the processor abstraction comprises programming code, and wherein at least a portion of the programming code for the processor abstraction is common as between at least

some processor blocks and is shared by such processor blocks (see, for example, column 10, lines 29-40, which shows that the abstractions include common programming code that is shared and reused), and

(h) wherein the programming code for the processor abstraction is organized into a tree form with generic code at a base node and more specific levels of code branching out at nodes therefrom, the processor abstraction nodes including leaf nodes from which no other nodes branch out, each processor block being defined to include a plurality of nodes extending from the base node to a particular leaf node (see, for example, FIG. 9 and column 27, lines 13-24, which shows that the abstractions are organized into trees with common code at the base node and derived code at the leaf nodes, and see, for example, column 28, lines 21-55, which shows one such block defined to include a plurality of nodes extending from the base node to a leaf node).

You illustrates a processor abstraction to provide memory addressing for each processor (see, for example, FIG. 9 and column 27, lines 1-7), but does not expressly illustrate the corresponding debugging type abstraction.

However, Niemi expressly discloses a debugging type abstraction as recited in the claim that is organized into a tree and provides debugging type services (see, for example, FIG. 4 and column 8, lines 11-42).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement You with a corresponding debugging type abstraction, such as taught by Niemi, so as to expressly represent the debugging types that the engine supports (see, for example, You, column 6, lines 31-55), as already intended (see, for example, You, column 5, lines 43-55). Furthermore, it would have been obvious to one of ordinary skill in the art at the

time the invention was made to implement the debugging type abstraction in the same manner as the processor abstraction of You, and to organize the debugging type abstraction into a corresponding tree (see, for example, You, FIG. 9).

With respect to claim 3 (currently amended), the rejection of claim 1 is incorporated, and You further discloses the limitation wherein the debugging services include services selected from a group comprising accessing memory, accessing context, accessing system information, inserting a breakpoint, removing a breakpoint, controlling execution, and combinations thereof (see, for example, column 10, lines 20-27, which shows services such as stack or memory access, runtime or context information, breakpoints, and so on).

With respect to claim 7 (currently amended), the rejection of claim 1 is incorporated, and You further discloses the limitation wherein the processor services include services selected from a group comprising recognizing particular processor instructions, recognizing processor states, maintaining hardware breakpoints, assembling code for the processor, disassembling code from the processor, disassembling code from a dump file produced by the processor, and combinations thereof (see, for example, column 10, lines 20-27, which shows services such as register or processor state information, breakpoints, hardware exceptions, and so on).

With respect to claim 10 (original), the rejection of claim 1 is incorporated, and You further discloses the limitation wherein the engine further includes a high level portion for issuing generic requests to the selected debugging type block and to the selected processor block to accomplish debugging actions (see, for example, FIG. 2 and column 9, lines 28-45, which

shows the high-level debugging interface for issuing generic requests to the selected classes or blocks of the engine).

With respect to claim 11 (original), the rejection of claim 10 is incorporated, and You further discloses the limitation wherein the plurality of debugging type blocks are organized into a debugging type abstraction available to provide debugging type services that vary in implementation for each debugging type, wherein the plurality of processor blocks are organized into a processor abstraction available to provide processor services that vary in implementation for each processor, and wherein the high level portion issues generic request to the debugging type abstraction and to the processor abstraction to accomplish debugging actions (see, for example, see, for example, column 5, lines 43-59, which shows that the engine is organized into abstractions to provide debugging services that vary in implementation, and see, for example, FIG. 2 and column 9, lines 28-45, which shows the high-level debugging interface for issuing generic requests to the selected classes or blocks of the engine).

With respect to claim 14 (original), the rejection of claim 1 is incorporated, and You further discloses the limitation wherein the plurality of processor attributes supported by the processor blocks include processor attributes representative of members selected from a group consisting of an X86 processor family, an ALPHA processor family, and IA64 processor family, and combinations thereof (see, for example, FIG. 9, which shows support for X86 and 64-bit processor families, among others).

With respect to claim 15 (original), the rejection of claim 1 is incorporated, and You further discloses the limitation wherein the debugger further has an executable for being

executed by a user, for calling the engine, and for providing an interface between the user and the engine (see, for example, column 9, lines 28-45, which shows the client interface executed by a user for calling the engine).

With respect to claim 18 (original), the rejection of claim 1 is incorporated, and You further discloses the limitation wherein the particular debuggee is a dump file produced by a processor operating in a particular mode, wherein the debugging type attribute of the dump file corresponds to the particular mode, and wherein the particular debugging type block of the engine selected for debugging the dump file supports the debugging type attribute of the dump file (see, for example, column 6, lines 31-55, which shows the debugging types supported by the engine, including postmortem debugging for inspecting the state of a program after it terminates, such as with a dump file that would identify the processing mode, such as with an attribute).

With respect to claim 19 (original), the rejection of claim 1 is incorporated, and You further discloses the limitation wherein the particular debuggee is a dump file produced by a type of processor, wherein the processor attribute of the dump file corresponds to the type of processor, and wherein the particular processor block of the engine selected for debugging the dump file supports the processor attribute of the dump file (see, for example, column 6, lines 31-55, which shows the debugging types supported by the engine, including postmortem debugging for inspecting the state of a program after it terminates, such as with a dump file that would identify the type of processor, such as with an attribute, and see, for example, column 9, lines 17-25, which shows a plurality of processor attributes supported by the engine).

With respect to claim 20 (previously presented), the limitations recited in the claim are analogous to those of claim 1 (see the rejection of claim 1 above).

With respect to claim 27 (original), the limitations recited in the claim are analogous to those of claim 10 (see the rejection of claim 10 above).

With respect to claim 28 (original), the limitations recited in the claim are analogous to those of claim 11 (see the rejection of claim 11 above).

With respect to claim 29 (original), the limitations recited in the claim are analogous to those of claim 15 (see the rejection of claim 15 above).

5. Claims 16, 17, 30 and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over You in view of Niemi, as applied to claims 15 and 29 above, respectively, and further in view of U.S. Patent No. 5,533,192 to Hawley et al. (art of record, "Hawley").

With respect to claim 16 (original), the rejection of claim 15 is incorporated, but although You discloses support for a plurality of debugging types (see, for example, column 6, lines 31-55), You does not expressly disclose the limitation wherein the executable includes an attribute that results in the selection of a particular debugging type block in the engine.

However, Hawley discloses an attribute in the executable used to select a particular debugger (see, for example, information 811 in FIG. 8A, and see, for example, step 853 in FIG. 8B and column 19, lines 12-22), in a debugging system having a plurality of debuggers (see, for example, the abstract).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement You with an attribute to select the appropriate debugger or debugging type block, such as taught by Hawley, in the absence of an alternative selection by the user (see, for example, Hawley, column 18, lines 26-37).

With respect to claim 17 (original), the rejection of claim 15 is incorporated, but although You discloses support for a plurality of processor families or types (see, for example, column 9, lines 17-26), You does not expressly disclose the limitation wherein the executable includes an attribute that results in the selection of a particular processor block in the engine.

However, Hawley discloses an attribute in the executable used to select a particular debugger (see, for example, information 811 in FIG. 8A, and see, for example, step 853 in FIG. 8B and column 19, lines 12-22), in a debugging system having a plurality of debuggers (see, for example, the abstract).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement You with an attribute to select the appropriate debugger or processor block, such as taught by Hawley, in the absence of an alternative selection by the user (see, for example, Hawley, column 18, lines 26-37).

With respect to claim 30 (original), the limitations recited in the claim are analogous to those of claim 16 (see the rejection of claim 16 above).

With respect to claim 31 (original), the limitations recited in the claim are analogous to those of claim 17 (see the rejection of claim 17 above).

Conclusion

6. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

TUAN DAM
SUPERVISORY PATENT EXAMINER